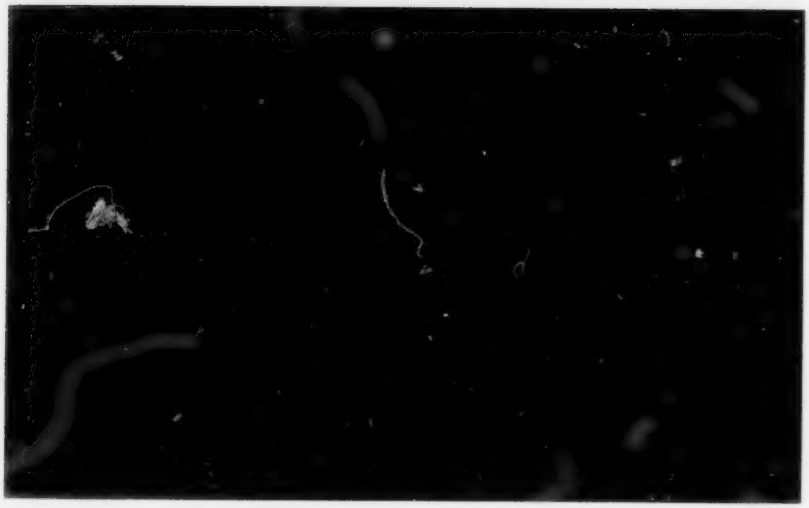
CENTRE DE RECHERCHE SUR LES TRANSPORTS
(C.R.T.)
CENTRE FOR RESEARCH ON TRANSPORTATION

UNIVERSITÉ DE MONTRÉAL
ÉCOLE DES HAUTES ÉTUDES COMMERCIALES
ÉCOLE POLYTECHNIQUE



CENTRE DE RECHERCHE SUR LES TRANSPORTS
UNIVERSITÉ DE MONTRÉAL
C.P. 6128, SUCCURSALE A,
MONTRÉAL
CANADA H3C 3J7

TÉLÉPHONE: (514) 343-7575
TÉLEX: (21) 05-561325
EMAIL: crt@crt.umontreal.ca
TÉLÉCOPIEUR (FAX): (514) 343-7121

91-04081/

CRT-763

Quelques problèmes de chemins pour les hypergraphes orientés

par
Lilian Markenzon*
Sang Nguyen

* En congé sabbatique de l'Université Fédérale de Rio de Janeiro. Ce travail a été réalisé grâce à une aide financière du CNQq (Conselho Nacional de Pesquisa e Desenvolvimento, Brasil), no.202137/89.9

Centre de recherche sur les transports — Publication #763

Avril 1991

Dépôt légal – 2ème trimestre 1991

RÉSUMÉ

Les BF-graphes sont une classe particulière d'hypergraphes orientés. Quelques applications diversifiées pour cette famille importante, comme les bases de données et l'intelligence artificielle, sont déjà connues. On peut montrer qu'ils peuvent aussi décrire le comportement des systèmes concurrents.

On présente ici l'approche théorique de quelques problèmes de cheminement pour les BF-graphes, surtout les BF-graphes acycliques. Après les concepts fondamentaux d'hypergraphes orientés, on présente l'algorithme pour la détermination d'un BF-chemin de complexité de $O(n \times size(\mathcal{E}))$; on discute aussi le problème de la couverture par des chemins et, enfin, on présente une solution polynomiale pour un cas particulier de deux problèmes de chemins avec contraintes, dans un BF-graphe acyclique.

Mots-clés: hypergraphes orientés, BF-graphes, chemins

I – Introduction

La représentation de la structure d'un programme par des graphes orientés est une approche fréquemment employée dans la validation des logiciels. Ntafos et Hakimi (1979) ont décrit un ensemble de problèmes issus de cette approche, parmis lesquels la couverture minimum par des chemins et la recherche de chemins satisfaisant des restrictions de paires impossibles et désirables. Les problèmes qui étaient proposés ont autant d'intérêt pratique que théorique. Ntafos et Hakimi (1981) proposaient l'utilisation des graphes structurés pour la description des programmes structurés, et présentaient de nouveaux résultats théoriques. Un résumé de ces résultats pour les graphes orientés acycliques est présenté dans Ntafos et Gonzalez (1984).

On travaille aussi avec les graphes pour étudier le comportement des systèmes concurrents (voir par exemple, Sahner et Trivedi (1987), El-Rewini et Lewis (1990), et Kapelnikov, Muntz et Ercegovac (1989)). Le niveau de détail varie de modèle en modèle: un sommet du graphe peut représenter une simple instruction de machine ou tout un processus entier. À ce moment-là, les BF-graphes, une classe particulière des hypergraphes orientés (Gallo et al. (1990)), peuvent jouer un rôle important, puisque ses arcs modèlent efficacement la simultanéité d'exécution des processus dans un environnement synchrone.

Kapelnikov, Muntz et Ercegovac (1989), présentent un nouveau modèle pour évaluer l'efficacité des systèmes concurrents qui est utilisé pour l'estimation du temps moyen d'exécution de l'implantation parallèle d'un programme. Pour ce calcul, on a définit le *computation control graph* (CCG), où les sommets représentent des tâches et les arcs modèlent des relations de précédence parmis ces tâches. L'existence de règles pour la classification de différents types d'arcs dans un CCG, pour refléter l'exécution simultanée des tâches précédentes ou non, nous indique une intéressante application pour les BF-graphes.

Cette nouvelle application justifie l'approche théorique de quelques problèmes de cheminement pour des BF-graphes. On présente ici l'étude de ces problèmes pour les BF-graphes acycliques. Notre objectif principal est de développer des algorithmes polynomiaux pour ces problèmes, et aussi de faire la comparaison de ces résultats avec les résultats connus pour le cas particulier des graphes orientés.

La section II de ce travail présente les concepts fondamentaux de hypergraphes orientés; la présentation de quelques problèmes de chemins pour les BF-graphes acycliques est faite aux

sections III et IV, et la section V offre des solutions polynomiaux pour les cas particuliers de problèmes avec des contraintes.

II – Définitions

Nous commençons par un rappel des hypergraphes orientés. La notation utilisée sera celle de Gallo et al. (1990).

Un hypergraphe est une paire $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ où $\mathcal{V}$ est l' ensemble de sommets, et où $\mathcal{E}$, $\mathcal{E} = \{e_1, e_2, \cdots, e_m\}$, $e_i \subseteq \mathcal{V}$ pour $i = 1, \ldots, m$, est l'ensemble d'hyper-arêtes.

Une hyper-arête orientée, ou *hyperarc*, $e = (L_1, L_2, \ldots, L_r)$ est une séquence ordonnée de r sous-ensembles de sommets, non vides et disjoints. Ces ensembles sont appelés les *couches* de l'hyperarc.

Un hyperarc sans couche est un *hyperarc vide*, alors qu'une *boucle* est un hyperarc avec une couche.

Étant donné l'hyperarc $e = (L_1, L_2, \cdots, L_r)$, les couches extrémales sont respectivement dénotées $tail(e) = L_1$ et $head(e) = L_r$.

Un *hypergraphe orienté* est un hypergraphe dont les arêtes sont orientées. Dorénavant on appelle les hypergraphes orientés simplement des hypergraphes.

Un *2-hypergraphe* $\mathcal{H}$, ou simplement, *2−graphe*, est un hypergraphe dont chaque hyperarc admet exactement deux couches. Les hyperarcs sont alors appelés *2-arcs*.

Un *B*-arc, est un 2-arc $e = (tail(e), head(e))$ où $|head(e)| = 1$ (figure 1a).

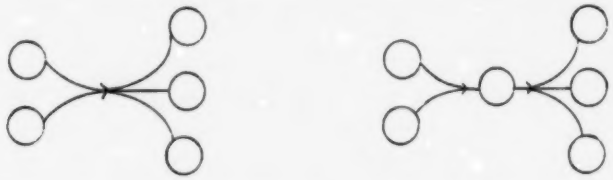
Un *F*-arc, est un 2-arc $e = (tail(e), head(e))$ où $|tail(e)| = 1$ (figure 1b).

figure 1

Un *B-graphe (F-graphe)* est un 2-graphe dont tous les hyperarcs sont des B-arcs (F-arcs). Un *BF-graphe* admet aussi bien les F-arcs que les B-arcs. Il est possible de tranformer un

2-graph en un BF-graphe en introduisant de nouveaux sommets. La figure 2 nous montre un 2-arc et sa transformation en deux arcs, un B-arc et un F-arc. Dans la suite un 2-arc sera simplement appelé un arc.

figure 2



On note $BS(v) = \{e \in \mathcal{E} \mid v \in head(e)\}$, et $FS(v) = \{e \in \mathcal{E} \mid v \in tail(e)\}$.

Soit un BF-graphe $\mathcal{H}$ avec n sommets et m arcs. On défine

$$size(\mathcal{E}) = \sum_{e \in \mathcal{E}} \left(|tail(e)| + |head(e)| \right)$$

Un *chemin* P_{st} dans un BF-graphe est une séquence de la forme

$$P_{st} = (V_1 = \{s\}, e_1, V_2, e_2, \cdots, e_q, V_{q+1} = \{t\})$$

où

- $V_1, V_2, \cdots, V_{q+1}$ sont des sous-ensembles de $\mathcal{V}$, appelés *ensembles d'articulation*

- $e_1, e_2, \cdots, e_q$ sont des arcs de $\mathcal{H}$

- $V_1 \subseteq tail(e_1), V_{q+1} \subseteq head(e_q)$, and $V_j \subseteq head(e_{j-1}) \cap tail(e_j)$, $j = 2, \ldots, q$

Un chemin P_{st} est *simple* si tous ses arcs sont différents; un chemin simple est *élémentaire* si tous ses ensembles d'articulation sont disjoints.

On dit que le sommet x est *connecté* au sommet y s'il y a un chemin P_{yx} en $\mathcal{H}$. La figure 3 présente un exemple de chemin; les ensembles d'articulation sont les sommets marqués.

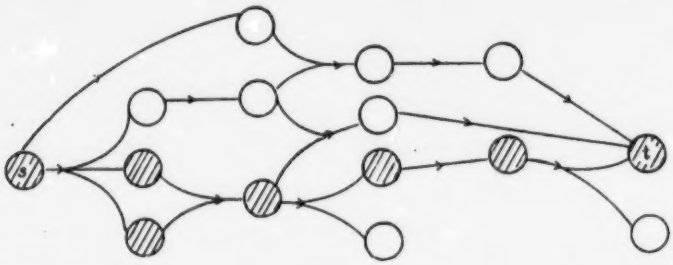
Si $s = t$, P_{st} est un *cycle*. Si le BF-graphe n'a pas de cycles, il est *acyclique*.

Soit le BF-graphe $\mathcal{H}$, acyclique. On appelle *niveau* du sommet x, $x \in \mathcal{V}$, par rapport à l'origine s, le nombre d' arcs au plus long chemin P_{sx}, la longueur de chaque arc étant 1.

III – Hyperchemins en BF-graphes

Dans un BF-graphe on peut distinguer trois types de chemins. Étant donné un BF-graphe $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, Gallo et al.(1990) distinguent le *B-chemin*, le *F-chemin* et le *BF-chemin*.

figure 3



Un *B-hyperchemin* π_{st} (ou *B-chemin*), d'origine s et de destination t, est un BF-graphe acyclique minimal pour l'inclusion $\mathcal{H}_\pi = (\mathcal{V}_\pi, \mathcal{E}_\pi)$ tel que:

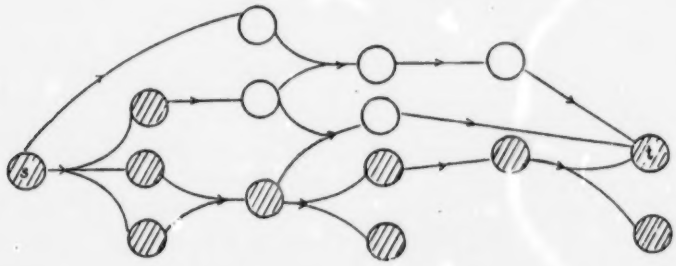
(i) $\mathcal{E}_\pi \subseteq \mathcal{E}$

(ii) $s, t \in \mathcal{V}_\pi = \bigcup_{e \in \mathcal{E}_\pi} e \subseteq \mathcal{V}$

(iii) $x \in \mathcal{V}_\pi \Rightarrow x$ est connecté à s dans $\mathcal{H}_\pi$

La figure 4 présente un B-chemin dans le BF-graphe de la figure 3.

figure 4



Un *F-hyperchemin* π_{st} (ou *F-chemin*), d'origine s et de destination t, est un BF-graphe acylique minimal pour l'inclusion $\mathcal{H}_\pi = (\mathcal{V}_\pi, \mathcal{E}_\pi)$ tel que:

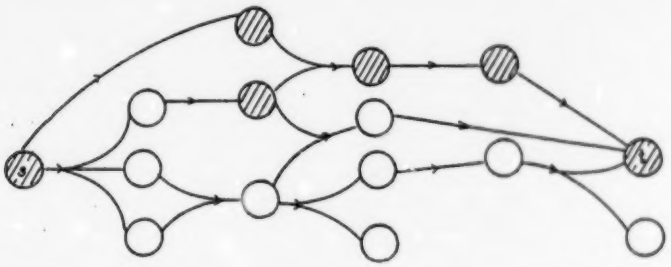
(i) $\mathcal{E}_\pi \subseteq \mathcal{E}$

(ii) $s, t \in \mathcal{V}_\pi = \bigcup_{e \in \mathcal{E}_\pi} e \subseteq \mathcal{V}$

(iii) $x \in \mathcal{V}_\pi \Rightarrow t$ est connecté à x dans $\mathcal{H}_\pi$

La figure 5 présente un B-chemin dans le BF-graphe de la figure 3.

figure 5



Un *BF-hyperchemin* π_{st} (ou *BF-chemin*), d'origine s et de destination t, est un BF-graphe acyclique minimal pour l'inclusion $\mathcal{H}_\pi = (\mathcal{V}_\pi, \mathcal{E}_\pi)$ tel que

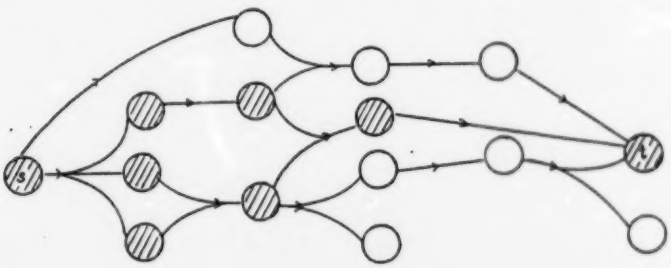
(i) $\mathcal{E}_\pi \subseteq \mathcal{E}$

(ii) $s, t \in \mathcal{V}_\pi = \bigcup_{e \in \mathcal{E}_\pi} e \subseteq \mathcal{V}$

(iii) $x \in \mathcal{V}_\pi \Rightarrow x$ est connecté à s et t est connecté à x dans $\mathcal{H}_\pi$

La figure 6 présente un B-chemin dans le BF-graphe de la figure 3.

figure 6



Un sommet x est *B-connecté* (*F-connecté, BF-connecté*) à un sommet y dans un BF-graphe, s'il existe un B-chemin, (F-chemin, BF-chemin) π_{yx} dans $\mathcal{H}$.

La procédure *Bvisit*$(s, \mathcal{H})$ (Gallo et al. (1990)) nous fournit un ensemble de B-chemins pour les sommets qui sont B-connectés à un sommet s, $s \in \mathcal{V}$ de $\mathcal{H}$. L'étiquette *blabel* indique, pour chaque sommet, s'il est visité ou non; l'étiquette *arrière* indique, pour chaque sommet, l'arc du B-chemin obtenu par la visite. De cette façon on peut reconstruire un B-chemin à partir de n'importe quel sommet de $\mathcal{H}$.

```
procedure Bvisit(s, H)
    pour i ∈ V faire
        blabel_i := faux;   arrière_i := 0
    fin
    pour e ∈ E faire
        t_e := 0
    fin
    Q := {s};   blabel_s := vrai
    tant que Q ≠ ∅ faire
        choisir i ∈ Q;   ôter i de Q
        pour e ∈ FS(i) faire
            t_e := t_e + 1;
            si t_e = |tail(e)| alors
                pour h ∈ head(e) | not blabel_h faire
                    Q := Q ∪ {h};   blabel_h := vrai;   arrière_h := e
                fin
            fin
        fin
    fin
```

On peut vérifier que, pour un BF-graphe, la complexité de $Bvisit(s, \mathcal{H})$ est $O(size(\mathcal{E}))$. En effet, pour chaque sommet v on examine les arcs $e \in FS(v)$. Si l'arc e est un B-arc, on l'examine au plus $|tail(e)|$ fois. Si l'arc e est un F-arc, cette étude sera faite une fois, parce que $|tail(e)| = 1$; mais $|head(e)|$ sommets seront mis dans l'ensemble Q. Cela nous donne $size(\mathcal{E})$ pour l' analyse de tous les arcs $e \in \mathcal{E}$.

Une F-visite pour l'hypergraphe $\mathcal{H}$ peut être établie de façon presque identique à la B-visite déjà présentée. Selon la définition de F-chemin on doit marquer les sommets x tels que t est connecté à x. Alors, on commence la visite par le sommet t. La procédure $Fvisit(t, \mathcal{H})$ nous fournit un ensemble de F-chemins pour les sommets qui sont F-connectés à t. L'étiquette *avant* nous permet ainsi de construire un F-chemin pour n'importe quel sommet de $\mathcal{H}$.

```
procedure Fvisit(s, H)
    pour i ∈ V faire
        flabel_i := faux;   avant_i := 0
    fin
    pour e ∈ E faire
        h_e := |head(e)|
    fin
    Q := {t};   flabel_t := vrai
```

```
tant que Q ≠ ∅ faire
    choisir i ∈ Q;   ôter i de Q
    pour e ∈ BS(i) faire
        h_e := h_e − 1
        si h_e = 0 alors
            pour y ∈ tail(e) | not flabel_y faire
                Q := Q ∪ {y};    flabel_y := vrai;    avant_y := e
            fin
        fin
    fin
fin
```

La complexité de la procédure $Fvisit(t, \mathcal{H})$ est aussi $O(size(\mathcal{E}))$.

L'ensemble Q peut être une file. Alors, pour le cas particulier où le BF-graphe est un graphe orienté, la procédure $Bvisit(s, \mathcal{H})$ nous présente une fouille en largeur et la procédure $Fvisit(t, \mathcal{H})$ une fouille en largeur inversée. La complexité traditionnelle est maintenue.

Pour établir un BF-chemin on introduit d'abord la notion de *graphe frontière*. Soit le BF-graphe $\mathcal{H} = (\mathcal{V}, \mathcal{E})$. On appelle un *graphe frontière* $\mathcal{H}' = (\mathcal{V}', \mathcal{E}', s, t)$ le BF-graphe où:

(i) $\mathcal{V}' = \{x \in \mathcal{V} \mid \exists$ B-chemin$_{sx}$ et $\exists$ F-chemins$_{xt}\}$

(ii) $\mathcal{E}' = \{e \in \mathcal{E} \mid \forall x \in tail(e), x \in \mathcal{V}'$ et $\forall y \in head(e), y \in \mathcal{V}'\}$

(iii) s est le sommet d'origine, t est le sommet de destination.

La figure 7 nous présente le BF-graphe $\mathcal{H}$ et son graphe frontière $\mathcal{H}'$.

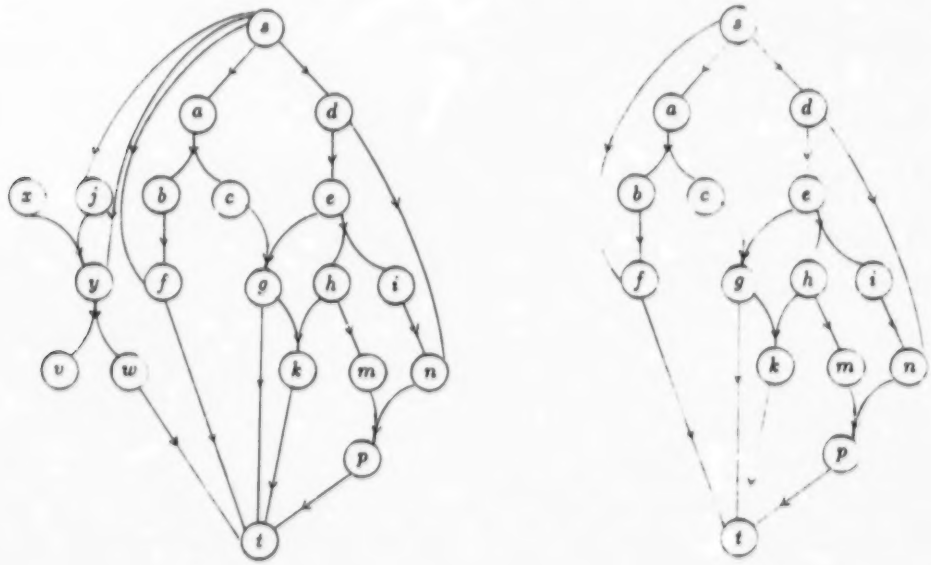
La définition de graphe frontière obéit aux conditions (i), (ii) et (iii) de la définition de BF-chemin, mais il peut ne pas être ni acyclique, ni minimal. Alors, un BF-chemin, s'il existe, est un sous-graphe de $\mathcal{H}'$.

La procédure $frontière(\mathcal{H}, \mathcal{H}', s, t)$ établit, pour un BF-graphe $\mathcal{H}$ donné, son graphe frontière $\mathcal{H}' = (\mathcal{E}', \mathcal{V}', s, t)$. À chacune itération on exécute les procédures *Bvisit* (à partir du sommet s) et *Fvisit* (à partir du sommet t). Les sommets x qui ne sont pas marqués par les deux procédures sont enlevés, parce qu'ils n'obéissent pas à la condition (i); on enlève aussi les arcs des ensembles $BS(x)$ et $FS(x)$. Les itérations sont arrêtées lorsqu'il ne reste plus de sommet à enlever. Si la construction est impossible, le graphe $\mathcal{H}'$ est vide.

```
procedure frontière(H, H', s, t)
    H' := H;    change := vrai
    tant que change faire
```

```
    change := faux
    pour v ∈ V' construire BS(v), FS(v)
    Bvisit(s, H'); Fvisit(t, H')
    pour v ∈ V' faire
        si not flabel_v or not blabel_v alors
            V' := V' − {v}; E' := E' − BS(v) − FS(v)
            change := vrai
        fin
    fin
    si s ∉ V' or t ∉ V' alors
        "construction impossible"; H' :=∅; change := faux
    fin
fin
```

figure 7



On connait déjà la complexité des deux procédures appelées. Dans le pire cas, seulement un sommet sera enlevé de $\mathcal{V}'$ chaque fois. Cela veut dire que l'on peut appeler les procédures $O(n)$ fois. La complexité de la procédure $frontière(\mathcal{H}, \mathcal{H}', s, t)$ est $O\big(n \times size(\mathcal{E})\big)$.

Si le BF-graphe est acyclique, une importante simplification peut être introduite. À ce moment-là on n'a plus besoin de refaire à chaque pas les procédures *Bvisit* et *Fvisit*. Il suffit de chercher les sommets $x \,|\, FS(x) = \emptyset$, parce que le sommet t n'est pas connecté à x,

et les sommets $x \mid BS(x) = \emptyset$, parce qu'ils ne sont pas connectés à s. Quand on enlève ces sommets, des nouveaux sommets qui obéissent à ces propriétés seront créés, et doivent être aussi enlevés. Cependant, chaque sommet, et ses arcs, ne sont analysés qu'une seule fois. Cette modification, qui réduit la complexité précédente par un facteur de n, est décrite ci dessous.

```
procedure A-frontière(H, H', s, t)
    H' := H;   Q := ∅
    pour e ∈ E faire
        edge_e := vrai
    fin
    pour v ∈ V faire
        nb_v := |BS(v)|;   nf_v := |FS(v)|;   node_v := vrai
        si (nb_v = 0 ∧ v ≠ s) or (nf_v = 0 ∧ v ≠ t) alors
            Q := Q ∪ {v};   node_v := faux;   V' := V' − {v}
        fin
    fin
    tant que Q ≠ ∅ faire
        choisir i ∈ Q;   ôter i de Q
        pour e ∈ BS(i) ∪ FS(i) and edge_e faire
            pour x ∈ tail(e) faire
                nf_x := nf_x − 1
                si nf_x = 0 and node_x alors
                    Q := Q ∪ {x};   node_x := faux;   V' := V' − {x}
                fin
            fin
            pour y ∈ head(e) faire
                nb_y := nb_y − 1
                si nb_y = 0 and node_y alors
                    Q := Q ∪ {y};   node_y := faux;   V' := V' − {y}
                fin
            fin
            E' := E' − {e};   edge_e := faux
            si s ∈ Q or t ∈ Q alors
                "construction impossible";   H' := ∅;   Q := ∅
            fin
        fin
    fin
```

La connaissance du graphe frontière permet d'établir un BF-chemin π_{st} dans un BF-graphe acyclique $\mathcal{H}$. On sait que, pour un graphe $\mathcal{H}$ acyclique, le graphe frontière est acyclique et satisfait les conditions (i), (ii) et (iii) de la définition de BF-chemin; toutefois il n'est

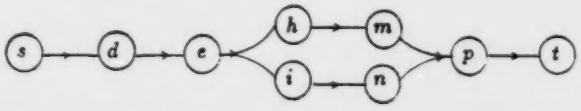
pas nécessairement minimal. À cause de cela on essaie d'enlever d'abord un sommet x (on obtient un nouveau graphe). Si on peut construire un graphe frontière pour ce nouveau graphe, le sommet x est superflu. On va répéter cet essai pour tous les sommets pour trouver un graphe frontière minimal par rapport aux sommets. Après avoir enlevé de la même façon les arcs superflus, le résultat est un BF-chemin. Il faut observer que ce BF-chemin n'est pas nécessairement unique, dû au fait qu'on peut obtenir des différents graphes frontières minimaux selon l'ordre suivant lequel on enlève les sommets. Ces graphes frontières sont les BF-chemins possibles.

La procédure $BFchemin(\mathcal{H})$ nous fournit un BF-chemin dans un BF-graphe acyclique $\mathcal{H}$:

procedure $BFchemin(\mathcal{H})$
    $A\text{-}frontière(\mathcal{H}, \mathcal{P}, s, t)$
    pour $x \in \mathcal{V}$ and $x \neq s$ and $x \neq t$ **faire**
        $\mathcal{P}' := \mathcal{P}$
        $\mathcal{P}' := \mathcal{P}' - \{x\}$
        $A\text{-}frontière(\mathcal{P}', \mathcal{P}'', s, t)$
        si $\mathcal{P}'' \neq \emptyset$ **alors**
            $\mathcal{P} := \mathcal{P}''$
        fin
    fin
    enlever les arcs superflus

Le BF-graphe $\mathcal{P} = (\mathcal{V}_\mathcal{P}, \mathcal{E}_\mathcal{P})$ désigne un BF-chemin de $\mathcal{H}$. La figure 8 présente le BF-chemin $\mathcal{P}$, résultat de $BFchemin(\mathcal{H})$ pour le BF-graphe de la figure 7.

figure 8



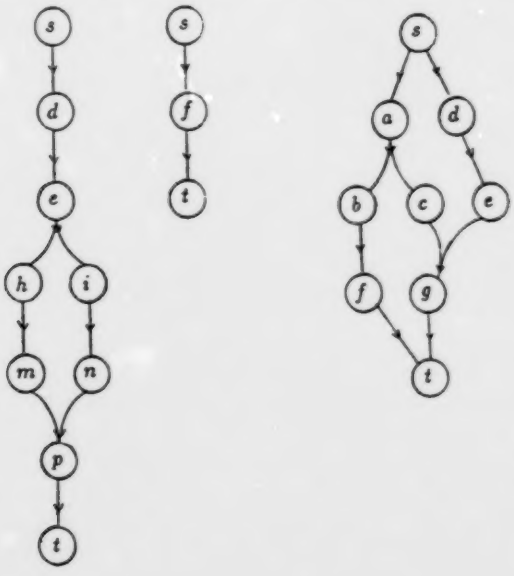
Pour déterminer la complexité de $BFchemin$ on observe que la procédure $A\text{-}frontière$ est appelée $O(n)$ fois. Alors, la complexité de l'algorithme est $O(n \times size(\mathcal{E}))$.

IV – Autres problèmes de BF-chemins

Une variation intéressante est le problème du BF-chemin qui passe par un ensemble de sommets $\mathcal{V}^*$, $\mathcal{V}^* \subseteq \mathcal{V}$. Ce problème, dont on connait une solution linéaire pour les graphes orientés due à Gabow et al. (1976), est aussi soluble pour les BF-graphes acycliques. Une modification dans l'algorithme *BFchemin* nous permettra de résoudre le problème. À chaque pas la procédure essaie d'enlever un sommet; comme l'ensemble $\mathcal{V}^*$ doit appartenir à l'ensemble solution, il faut que les sommets de $\mathcal{V}^*$ soient les derniers examinés. Alors, il est suffisante de faire d'abord l'exécution pour l'ensemble $\mathcal{V} - \mathcal{V}^*$; on vérifie ensuite, suivant le même principe, si chaque sommet de $\mathcal{V}^*$ doit ou non rester dans $\mathcal{P}$. Si la réponse est affirmative, $\mathcal{P}$ est le BF-chemin désiré. La complexité du nouveau algorithme est la même que la complexité de *BFchemin*.

Un autre problème très connu, et qui présente un résultat intéressant pour les BF-graphes, est le problème de la recherche d'une *couverture minimum par des chemins* : trouver un ensemble de chemins $P = \{p_1, p_2, \cdots, p_q\}$, d'origine s et de destination t, tel que chaque sommet $v \in V$ appartient à un chemin p_j; cet ensemble doit être de cardinalité minimum. Ntafos et Hakimi (1979) ont présenté des résultats polynomiaux pour ce problème, pour les graphes orientés acycliques et les graphes structurés.

figure 9



Pour les BF-graphes acycliques on observe qu'il peut exister des sommets du graphe frontière n'appartenant à aucun BF-chemin du BF-graphe. La figure 9 nous présente tous les BF-chemins possibles pour le BF-graphe de la figure 7. Il n'y a aucun chemin passant par le sommet k, donc la couverture minimum est vide.

V – BF-chemins avec des contraintes

V.1 – Les problèmes

Soit un graphe orienté $G = (V, E)$ et un ensemble C de paires (x, y), $x, y \in V$. Deux problèmes peuvent être posés:

Problème du chemin avec restriction des paires impossibles (CRPI): Existe-t-il un chemin de l'origine s à la destination t en G qui contient au maximum un sommet de chaque paire de C?

Problème du chemin avec restriction des paires désirables (CRPD): Existe-t-il un chemin de l'origine s à la destination t tel que, pour chaque paire $(a, b) \in C$, si le chemin contient a il contient aussi b?

On connait quelques résultats pour ces problèmes. Gabow et al. (1976) ont montré que le problème des paires impossibles est NP-complet pour les graphes orientés acycliques. Le problème des paires désirables est aussi NP-complet, même pour les graphes acycliques structurés (Ntafos et Hakimi (1981)). On connait la solution polynomiale seulement pour les arbres. Cependant, pour des cas particuliers de paires on a réussi à trouver une solution polynomiale (Markenzon et Szwarcfiter (1987)).

On généralise ici les problèmes pour les BF-chemins en BF-graphes acycliques, en présentant une solution polynomiale pour un cas particulier de l'ensemble de paires C. Cette solution est meilleure pour les graphes orientés que la solution déjà connue.

Soit $\mathcal{H} = (\mathcal{V}, \mathcal{E}, s, t)$ un graphe frontière. Soit $a, b \in \mathcal{V}$, tels que $niveau(a) \leq niveau(b)$. On dit que le sommet a est *fort par rapport au sommet* b si le sommet a appartient à tout B-chemin$_{sb}$. On dit que le sommet b est *fort par rapport au sommet* a si le sommet b appartient à tout F-chemin$_{at}$. Si, dans la paire (a, b) seulement un sommet est fort, l'autre sera appelé *faible*.

Soit $\mathcal{F}_{xy}$ l'ensemble des F-chemins$_{xy}$ et $\mathcal{B}_{xy}$ l'ensemble des B-chemins$_{xy}$. Dans un graphe

frontière, on observe que

$$\mathcal{F}_{sb} \subset \mathcal{B}_{sb}$$

$$\mathcal{B}_{at} \subset \mathcal{F}_{at}$$

On dit qu'une paire de sommets (a, b) est *stable* si au moins un des sommets est fort ou si $\mathcal{B}_{sb} \cap \mathcal{F}_{at} = \emptyset$.

On a déjà vu que les problèmes CRPD et CRPI sont NP-complets pour les graphes acycliques et même pour des cas très particuliers, comme les graphes structurés; ils sont solubles en temps polynomial seulement pour les arbres (Gabow et al. (1976)). Cependant, on peut voir d'une façon intuitive, que, si on se rapporte aux paires stables, les problèmes peuvent aussi être résolus en temps polynomial. Dans une paire stable, par définition, tous les chemins qui passent par le sommet faible, passent aussi par le sommet fort. Ainsi, le sommet faible doit être transformé de telle façon que les deux sommets deviennent forts ou l'ensemble de chemins parmis les deux sommets devient vide. Cette solution est obtenue simplement en enlevant des arcs qui génèrent des chemins indésirables. On réussit de cette façon à trouver des chemins uniques.

Ainsi, on peut montrer, comme dans Markenzon et Szwarcfiter (1987), que:

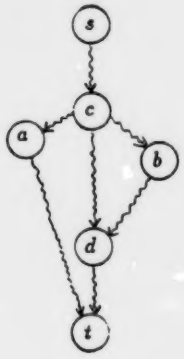
I) Si une paire (a, b) est stable en $\mathcal{H}$, elle l'est aussi en $\mathcal{H}'$, graphe frontière qu'on obtient en enlevant de $\mathcal{H}$ quelques arcs.

II) Les problèmes CRPD et CRPI sont solubles en temps polynomial si l'ensemble C est un ensemble de paires stables.

V.2 – L'algorithme

La résolution des problèmes CRPD et CRPI pour un ensemble de paires stables est simple. Il faut seulement trouver les paires de C qui sont stables et exécuter les procédures propres à chaque problème. En général, cela veut dire enlever les arcs qui font partie des chemins indésirables. Il faut remarquer qu'une paire non stable peut la devenir à cause des actions prises pour la solution d'une autre paire. La figure 10 présente un exemple pour le problème CRPD, pour l'ensemble $C = \{(a, b), (c, d)\}$. Il n'y a pas de chemin possible pour la paire (a, b); il faut alors enlever $BS(a)$ et $BS(b)$. La paire (c, d) devient alors stable et le problème a une solution. À cause de cela on doit refaire ces tests jusqu'à l'épuisement de l'ensemble C.

figure 10



L'algorithme CRP présente, comme résultat, un graphe frontière; n'importe quel BF-chemin dans ce graphe satisfait le problème. Pour chacune des paires (a, b) de l'ensemble C on détermine $\mathcal{F}_{at}$ et $\mathcal{B}_{sb}$; ensuite il faut tester si la paire est stable ou non. Si la paire est stable, on prend les mesures adéquats pour le probème désiré et on enlève (a, b) de C. La procédure est répeté jusqu'à ce que l'ensemble C devient vide ou contient seulement des paires non stables. Voici l'algorithme CRP:

```
algorithme CRP
    change := vrai
    tant que change faire
        change := faux
        pour (a,b) ∈ C faire
            determination de F_at et B_sb        (1)
            test(a,b,type)                       (2)
            si type = stable alors
                change := vrai
                C := C − {(a,b)}
                CRPD or CRPI                     (3)
            fin
        fin
fin
```

Il faut détailler trois parties dans l'algorithme CRP. D'abord, la détermination de $\mathcal{F}_{at}$ (1). Pour cela on exécute une fouille inversée, à partir du sommet a. La procédure $Fset(a, t, \mathcal{H}, \mathcal{F})$, présentée ici, établit deux résultats:

- l'ensemble de chemins $\mathcal{F}_{at} = (\mathcal{V}_{\mathcal{F}}, \mathcal{E}_{\mathcal{F}})$

– l'ensemble de sommets $\widehat{\mathcal{V}}_{\mathcal{F}} = \{x \mid$ existe un chemin $P_{ax}\}$.

On note que $\widehat{\mathcal{V}}_{\mathcal{F}} \subseteq \mathcal{V}_{\mathcal{F}}$.

```
procedure Fset(a, t, H, F)
    pour x ∈ V faire
        fmark_x := faux
    fin
    F := ∅
    Q := {a};   fmark_a := vrai
    tant que Q ≠ ∅ faire
        choisir i ∈ Q;   ôter i de Q
        pour e ∈ FS(i) faire
            si e ∉ E_F alors
                E_F := E_F ∪ {e}
                V_F := V_F ∪ {x | x ∈ tail(e) or x ∈ head(e)}
                pour x ∈ head(e) and not fmark_x faire
                    Q := Q ∪ {x};   fmark_x := vrai
                fin
            fin
        fin
    fin
```

La procédure $Bset(s, b, \mathcal{H}, \mathcal{F})$, étant symétrique à la procédure $Fset$, établit aussi deux résultats:

– l'ensemble de chemins $\mathcal{B}_{sb} = (\mathcal{V}_{\mathcal{B}}, \mathcal{E}_{\mathcal{B}})$

– l'ensemble de sommets $\widehat{\mathcal{V}}_{\mathcal{B}} = \{x \mid$ existe un chemin $P_{xb}\}$.

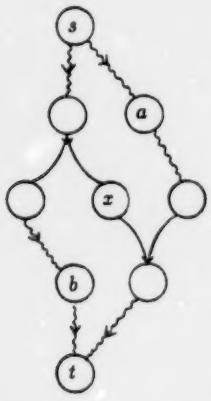
Deuxièmement, notez dans l'algorithme CRP le test pour vérifier si la paire (a, b) est stable ou non. On examine d'abord si $\mathcal{F}_{at} \cap \mathcal{B}_{sb} = \emptyset$; le résultat du test est alors imédiat. Si $\mathcal{F}_{at} \cap \mathcal{B}_{sb} \neq \emptyset$, on examine les sommets x, $x \in (\widehat{\mathcal{V}}_{\mathcal{F}} \cap \widehat{\mathcal{V}}_{\mathcal{B}})$. Ces sommets sont les sommets qui appartiennent aux chemins de a à b; si les arcs qui arrivent à x n'appartiennent pas à $\mathcal{E}_{\mathcal{F}}$ cela veut dire que le sommet a n'est pas fort, parce que cet arc établit un chemin alternatif à partir de s. On peut raisonner de façon symétrique pour les arcs qui partent de x. La procédure *test* présente la solution pour ce problème.

```
procedure test(a, b, type)
    Atype := vrai;   Btype := vrai
    type := nostable;   vertex := faux
    si F_at ∩ B_sb = ∅ alors
        type := stable
```

```
sinon
      pour x ∈ V | bmark_x = fmark_x = vrai faire
            vertex := vrai
            pour e ∈ FS(x) and x ≠ b faire
                  si e ∉ E_B alors Btype := faux
                  fin
            fin
            pour e ∈ BS(x) and x ≠ a faire
                  si e ∉ E_F alors Atype := faux
                  fin
            fin
fin
si vertex and (Atype or Btype) alors type := stable
fin
```

La variable *vertex* sert à signaler un cas particulier important. Si $\mathcal{F}_{at} \cap \mathcal{B}_{sb} \neq \emptyset$ et $\widehat{\mathcal{V}}_{\mathcal{F}} \cap \widehat{\mathcal{V}}_{\mathcal{B}} = \emptyset$, la paire (a, b) n'est pas stable. La figure 11 nous montre un exemple de ce cas.

figure 11



Finalement, la troisième partie de l'algorithme CRP demande des procédures spéciales pour chacun des problèmes, CRPD ou CRPI. Pour le problème CRPD on doit enlever des arcs de tel façon que, dans les paires stables, les deux sommets deviennent forts. L'autre cas est si $\mathcal{F}_{at} \cap \mathcal{B}_{sb} = \emptyset$: cela veut dire qu'il n'existe pas un chemin qui passe par a et b. Alors, ces deux sommets doivent disparaître du graphe. Pour le problème CRPI, on veut que la

situation soit inversée. Alors, on doit enlever les arcs de tel façon que le sommet faible dans les paires stables disparaîsse (on maintient ainsi les chemins qui passent seulement par le sommet fort). Si $\mathcal{F}_{at} \cap \mathcal{B}_{sb} = \emptyset$, le problème a déjà une solution.

On présente ici le résumé des actions à suivre pour chaque cas:

Problème CRPD:

- $\mathcal{F}_{at} \cap \mathcal{B}_{sb} = \emptyset$

  Action: $\mathcal{E} := \mathcal{E} - BS(a) - BS(b)$

- la paire (a, b) est stable

  - a et b sont des sommets forts

    Action: aucune

  - un de sommets est faible

    Action:

    $\mathcal{E} = \mathcal{E} - \{e \mid \exists x \in tail(e), bmark_x = fmark_x = vrai, e \notin \mathcal{E}_B\}$, si b est le sommet faible

    $\mathcal{E} = \mathcal{E} - \{e \mid \exists x \in head(e), bmark_x = fmark_x = vrai, e \notin \mathcal{E}_F\}$, si a est le sommet faible

Problème CRPI:

- $\mathcal{F}_{at} \cap \mathcal{B}_{sb} = \emptyset$

  Action: aucune

- la paire (a, b) est stable

  - a et b sont des sommets forts

    Action: $BS(a) := \emptyset$; $\quad BS(b) := \emptyset$

  - un des sommets est faible

    Action: $BS(x) := \emptyset$, pour le sommet faible x

Après chacune de ces actions, le BF-graphe $\mathcal{H}$ doit être remplacé par son graphe frontière, si ce dernier existe. Sinon, le chemin désiré est impossible.

V.3 – Complexité de l'algorithme CRP

L'algorithme doit essayer toutes les paires de C à chaque exécution de la commande **tant que**, c'est à dire $O(k^2)$ fois, pour $|C| = k$. Pour chacune des paires on doit refaire le graphe frontière d'un BF-graphe acyclique, ce qui nous donne $O\big(k^2 \times size(\mathcal{E})\big)$.

Bien sûr, si l'ensemble C est composé seulement de paires stables, cette complexité peut être réduite à $O\big(k \times size(\mathcal{E})\big)$, parce que le parcours par l'ensemble C sera fait seulement une fois.

Pour le cas particulier d'un graphe orienté, où $size(\mathcal{E}) = O(m)$, le graphe frontière peut être calculé en $O(m)$. Alors, l'algorithme CRP devient $O(k^2 m)$. Si $k < n$, le résultat est encore mieux que le résultat de Markenzon et Szwarcfiter (1987), où la complexité était $O(k\,n\,m)$ en utilisant d'autres structures de données.

VI – Conclusions

On a montré que l'on peut décrire un programme destiné à un environnement parallèle par les BF-graphes. On a ensuite examiné d'un point de vue théorique quelques problèmes de chemins dans les BF-graphes, notamment la détermination d'un BF-chemin, l'analyse du problème de la couverture minimum par des chemins et la détermination du BF-chemin qui passe par un ensemble de sommets $\mathcal{V}^*$, $\mathcal{V}^* \subseteq \mathcal{V}$. Finalement, on a présenté des solutions polynomiaux pour un cas particulier de l'ensemble de paires pour les problèmes avec contraintes de paires dans les BF-graphes acycliques.

Références

G. Gallo, G. Longo, S. Nguyen, S. Pallottino, "Directed Hypergraphs and Applications", *TR-3/90*, Dipartimento di Informatica, Univ. di Pisa, 1990

H.N. Gabow, S.N. Maheshwari, L.J. Osterweil, "On two problems in the generation of program test paths", *IEEE Trans. on Soft. Eng.*, SE-2, pp.227-231, 1976

A. Gibbons, W. Rytter, *Efficient Parallel Algorithms*, Cambridge University Press, 1988

A. Kapelnikov, R.R. Muntz, M.D. Ercegovac, "A Modeling Methodology for the Analysis of Concurrent Systems and Computations", *J. of Paral. and Dist. Comput.*, v.6, pp.568-597, 1989

L. Markenzon, J.L. Szwarcfiter, "Dois problemas de caminhos com restrições", XX Simpósio Brasileiro de Pesquisa Operacional, pp.142, 1987

S.C. Ntafos, T. Gonzalez, "On the computational complexity of path cover problems", *J. of Comp. and Syst. Sc.*, v.29, pp.225-242, 1984

S.C. Ntafos, S.L. Hakimi, "On path cover problems in digraphs and applications to program testing", *IEEE Trans. on Soft. Eng.*, SE-5, pp.520-529, 1979

S.C. Ntafos, S.L. Hakimi, "On structured digraphs and problem testing", *IEEE Trans. Comput.*, C-30, pp.67-77, 1981

H. El-Rewini, T.G. Lewis, "Scheduling Parallel Program Tasks onto Arbitrary Target Machines", *J. of Paral. and Dist. Comput.*, v.9, pp.138-153, 1990

R.A. Sahner, K.S. Trivedi, "Performance and Reliability Analysis Using Directed Acyclic Graphs", *IEEE Trans. on Soft. Eng.*, vol.SE-13, pp.1105-1114, 1987